K-фронтальный метод неравномерных покрытий

А.Ю.Горчаков

***Аннотация*—В данной работе предлагается параллельная реализация метода неравномерных покрытий, предназначенного для вычислительных систем с общей памятью. Метод неравномерных покрытий является одним из наиболее известных детерминированных методов решения задач глобальной оптимизации, основанный на схеме ветвей и границ. Из-за высокой вычислительной сложности метода и широкую доступность высокопроизводительных многоядерных систем с общей памятью, особую актуальность приобретает разработка параллельных реализаций данного метода. Существуют различные подходы к распараллеливанию метода неравномерных покрытий. Во-первых, существует несколько стандартов создания многопоточных приложений таких как OpenMP, MPI, функциональность C++17. Во-вторых, различаются подходы к организации хранения и обращения к спискам подзадач и синхронизации между потоками. В-третьих, различаются процедуры ветвления и вычисления оценок. Ранее проводилось сравнение указанных подходов к распараллеливанию метода неравномерных покрытий. Один из наиболее эффективных методов – фронтальный метод неравномерных покрытий, использует стратегию ветвления в ширину, для распараллеливания применяется технология OpenMP и структура хранения данных состоит из двух массивов пулов/списков подзадач. Указанный подход имеет следующие достоинства – простота реализации, достаточное быстродействие и стабильность работы. В качестве недостатков метода можно указать высокие требования к объему оперативной памяти. Для устранения данного недостатка в настоящей статье предлагается параллельная реализация К-фронтального метода неравномерных покрытий. Метод протестирован с использованием библиотеки тестовых функций на гибридном высокопроизводительном вычислительном комплексе ЦОД ФИЦ ИУ РАН.**

***Ключевые слова*—параллельные алгоритмы, метод неравномерных покрытий, OpenMP.**

I. Введение

Методы поиска глобального экстремума условно делятся на две группы – с доказательством (детерминированные) и без доказательства оптимальности (недетерминированные). К первой группе относятся различные варианты метода ветвей и границ (метод неравномерных покрытий [1]), методы интервального анализа [2] и многие другие. Вторая группа методов включает в себя различные стратегии случайного поиска [3-6], методы имитации отжига [7,8], генетические и роевые алгоритмы [9].

Обе группы методов, при решении практических задач глобальной оптимизации, требуют привлечения высокопроизводительной вычислительной техники. Типичные конфигурации вычислительных систем с общей памятью позволяют запускать на исполнение десятки и сотни одновременно работающих вычислительных потоков. При этом стандартный объем оперативной памяти у современных систем достигает от 0,5 до 1,5 терабайт.

В данной работе исследуется параллельная реализация фронтального метода неравномерных покрытий [10, 11, 12], в части требований к объему оперативной памяти. Для снижения требований к объему памяти предлагается его естественная модификация – К-фронтальный метод неравномерных покрытий.

Исследования и численные эксперименты проводились с использованием библиотеки тестовых функций [13, 14]. Для выполнения расчетов был использован гибридный высокопроизводительный вычислительный комплекс ФИЦ ИУ РАН [15].

II. Фронтальный метод неравномерных покрытий

Рассмотрим задачу оптимизации, в которой требуется найти минимум функции на заданном множестве

$$f(x) \to min, x \epsilon X$$

где X параллелепипед $\in R^n$. Идея метода неравномерных покрытий состоит в разбиении исходного множества X на подмножества с отсевом подмножеств, которые заведомо не содержат глобального минимума.

Параллельная реализация метода со стратегий ветвления в «ширину» (будем называть ее фронтальным методом) является достаточно простой в реализации. Распараллеливание производится с помощью директив OpenMP. Определяется значение thread_num – количество потоков, которое можно запустить одновременно. Для большинства архитектур, за редким исключением, оно берется равным количеству логических вычислительных ядер в системе (вызывается функция omp_get_thread_num()). Создается два массива пулов pool и pool_new, длина массивов thread_num. Чтение задач из массива пулов pool, распараллеливается с помощью директив #pragma omp parallel и #pragma

Работа выполнена при финансовой поддержке проекта РФФИ № 18-07-00566.

А.Ю. Горчаков – старший научный сотрудник Вычислительного центра им. А.А. Дородницина Федерального исследовательского центра «Информатика и управление» Российской академии наук. andrgor12@gmail.com.

omp for nowait schedule(dynamic), новые задачи каждый поток записывает в свой элемент массива pool_new. После исчерпания задач в массиве пулов pool и pool_new меняются местами (меняются ссылки на массивы). Синхронизация обновления рекордного значения производится с помощью механизма атомарных операций. Приведем формальное описание метода:

В основе алгоритма лежит процедура PBnB_front.

Общие переменные

r – рекордное значение

x – рекордное решение

procedure PBnB_front(L, ms)

L – массив списков подзадач

ms – максимальное (на выходе реальное) число шагов

```
1: s := 0
2: while pool is not empty and s < ms do
3:   s := s + size(pool)
4: #pragma omp parallel for
5:   for Q from pool
6:     if eval_omp(Q, r, x) = true then
7:       (r1, x1) = getSolution(Q)
8:       update_omp(r1, x1)
9:     else
10:        decomp Q into Q1, Q2
11:        pool_new[i].append(Q1)
12:        pool_new[i].append(Q2)
13:   endif
14: pool.swap(pool_new)
15: endwhile
16: ms := s
```

В пунктах 11,12 pool_new[i] элемент массива соответствующий i-му потоку вычисления.

Обновление рекордного значения и решения выполняется в процедуре update_omp.

procedure update_omp(rnew, xnew)

rnew – кандидат на рекордное значение

xnew – кандидат на рекордное решение

```
1: critical section
2: if r > rnew
3:   r = rnew
4:   x = xnew
5: end critical section
```

В функции update_omp используется механизм критических секций OpenMP (#pragma omp critical). В процедуре eval_omp используется атомарное чтение, реализованное директивой #pragma omp atomic read.

Для оценки необходимого объема оперативной памяти проведен численный эксперимент. Для 12 тестовых функций из библиотеки [13, 14] PBnB_front запущен 100 раз и вычислено среднее количество решенных подзадач (количество итераций), среднее от максимального количества подзадач находящихся в массивах pool и pool_new, среднее от максимального количества оперативной памяти, зарезервированной под массивы pool и pool_new. Оперативная память в целях сопоставления измерялась не в мегабайтах, а в количестве подзадач. Результаты приведены в таблице 1.

Таблица 1. Среднее количество итераций, максимальное количество подзадач, хранящихся в оперативной памяти, и объем зарезервированной памяти для подзадач для метода PBnB_front.

Функция	Кол-во итераций	Кол-во подзадач в памяти	Объем памяти
Dolan	1 180 809	537 006	1 209 457
Goldstein Price	2 380 541	655 881	1 341 071
Hartman 6	1 436 757	289 374	632 223
Hosaki	4 838 256	1 617 560	3 572 797
Jennrich-Sampson	459 424	145 932	296 973
Mishra 9	1 000 643	584 596	1 114 481
Powell Singular 2_8s	744 647	148 619	317 074
Quadratic	1 114 037	354 249	779 438
Shubert 2	479 751	148 354	306 775
Styblinski-Tang	542 672	173 639	382 533
Trecanni	813 093	263 676	553 789
Wayburn Seader 3	771 226	240 068	494 533

Как видно из таблицы 1 необходимый объем памяти составляет в среднем 70% от количества итераций. Для удобства процентные соотношения приведены на рисунке 1.

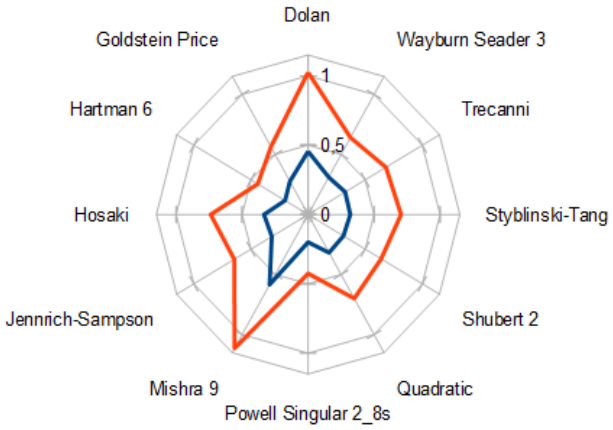

Рис. 1. Отношение количества подзадач и объема памяти к количеству итераций для метода PBnB_front.

Дадим оценку необходимого объема памяти в мегабайтах:

- Пусть размерность задачи n=6
- Количество итераций (число шагов) = 1 000 000
- Расчеты ведутся с двойной точностью

Тогда для хранения списка подзадач будет зарезервировано

2*n*sizeof(double)*<количество итераций>*0.7 = 64Мб

Учитывая, то что 1Тб уже является некоторым стандартом, для большинства задач ситуация с

нехваткой оперативной памяти представляется маловероятной.

Тем не менее, можно подобрать тестовые задачи (например, Trid 6, Trid 10), для которых риск нехватки оперативной памяти реализуется. Для устранения данного риска был предложен метод с комбинированной стратегией ветвления.

III. К-ФРОНТАЛЬНЫЙ МЕТОД НЕРАВНОМЕРНЫХ ПОКРЫТИЙ

Идея метода заключается в том, что в цикл ветвления в «ширину» добавляется к-шагов поиска в «глубину».

Приведем формальное описание метода:

В основе алгоритма лежит процедура PBnB_kfront.

Общие переменные

r – рекордное значение

x – рекордное решение

procedure PBnB_kfront(L, ms, K)

L – массив списков подзадач

ms – максимальное (на выходе реальное) число шагов

K – количество шагов в «глубину»

```
1: s := 0
2: while pool is not empty and s < ms do
3:   s := s + size(pool)
4: #pragma omp parallel for
5:   for Q from pool
6:     if eval_omp(Q, r, x) = true then
7:       (r1, x1) = getSolution(Q)
8:       update_omp(r1, x1)
9:     else
10:        decomp Q into Q1, Q2
11:        pool_new[i].append(Q1)
12:        pool_new[i].append(Q2)
13:  endif
14:  if size(pool) >= thread_num then
15:    for k=1 to K
16:      get Q from pool_new[i]
17:      if eval_omp(Q, r, x) = true then
18:        (r1, x1) = getSolution(Q)
19:        update_omp(r1, x1)
20:      else
21:        decomp Q into Q1, Q2
22:        pool_new[i].append(Q1)
23:        pool_new[i].append(Q2)
24:      endif
25: pool.swap(pool_new)
26: endwhile
27: ms := s
```

Пункты 14-24 соответствуют поиску в глубину, в п.16 подзадача всегда берется из конца i-го списка подзадач, цикл пп.15-24 выполняется либо K раз, либо до полного исчерпания i-го списка подзадач. Запустим к-фронтальный метод для того же набора функций с параметром K=1. Результаты запуска приведены в таблице 2 и на рисунке 2.

Таблица 2. Среднее количество итераций, максимальное количество подзадач, хранящихся в оперативной памяти, и объем зарезервированной памяти для подзадач для метода PBnB_kfront.

Функция	Кол-во итераций	Кол-во подзадач в памяти	Объем памяти
Dolan	38 389	6 717	13 973
Goldstein Price	2 380 435	343 460	704 379
Hartman 6	1 321 205	178 228	365 558
Hosaki	4 838 276	957 323	2 073 907
Jennrich-Sampson	459 442	109 562	250 239
Mishra 9	711 131	426 151	828 835
Powell Singular 2_8s	243 919	55 949	113 553
Quadratic	1 114 055	240 713	516 511
Shubert 2	478 340	99 310	242 008
Styblinski-Tang	542 708	128 387	257 411
Trecanni	813 093	168 066	358 771
Wayburn Seader 3	771 216	170 979	380 959

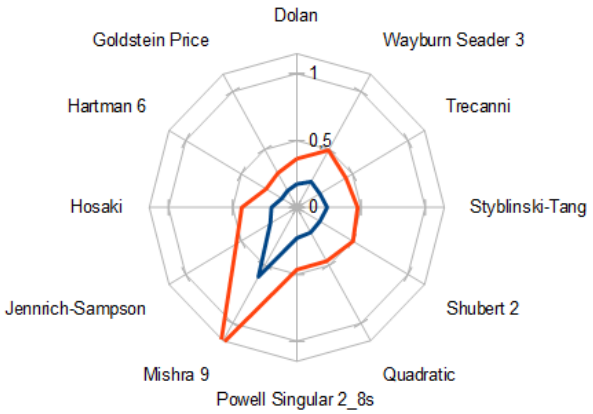

Рис. 2. Отношение количества подзадач и объема памяти к количеству итераций для метода PBnB_kfront.

Как мы можем увидеть, необходимый объем памяти снижен до 50% от количества итераций. Кроме этого, для некоторых функций, произошло существенное уменьшение количества самих итераций. На рисунке 3 приведен график зависимости объема памяти в зависимости от параметра K для двух тестовых функций – «Hosaki» и «Mishra 9».

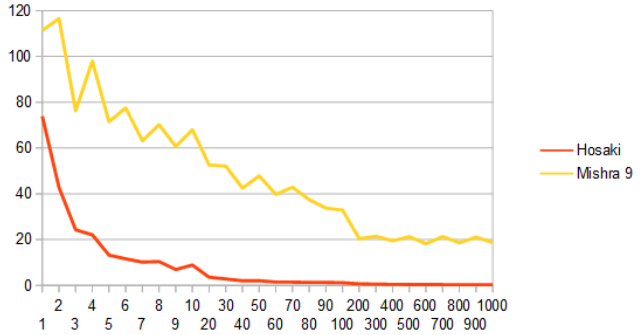

Рисунок 3. Объем памяти в зависимости от K (количества шагов поиска в «глубину») для метода PBnB_kfront.

Как видно из рисунка, для функции «Hosaki» объем памяти при увеличении K постоянно убывает, а для функции «Mishra 9» начиная с какого-то момента стабилизируется на уровне 20%.

IV. Заключение

Анализ предложенного в работе к-фронтального метода неравномерных покрытий показал, что использование комбинированной стратегии ветвления позволяет снизить требования к объему оперативной памяти. При этом схема метода не претерпела существенных изменений.

Библиография

[1] Евтушенко Ю. Г. Численный метод поиска глобального экстремума функций (перебор на неравномерной сетке) //Журнал вычислительной математики и математической физики, 1971, vol. 6. – pp.1390-1403.

[2] Hansen E., Walster G. W. Global optimization using interval analysis: revised and expanded. – CRC Press, 2003.

[3] Hickernell F. J., Yuan Y. A simple multistart algorithm for global optimization. – 1997.

[4] Martí R., Moreno-Vega J. M., Duarte A. Advanced multi-start methods //Handbook of metaheuristics. – Springer, Boston, MA, 2010. – C. 265-281.

[5] Martí R. et al. Multi-start methods //Handbook of Heuristics. – 2016. – C. 1-21.

[6] Pagès G. The Quasi-Monte Carlo Method //Numerical Probability. – Springer, Cham, 2018. – C. 95-132.

[7] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598), 671-680 (1983).

[8] Лопатин А. С. Метод отжига //Стохастическая оптимизация в информатике. – 2005. – Т. 1. – №. 1. – С. 133.

[9] П.В. Матренин, М.Г. Гриф, В.Г. Секаев, Методы стохастической оптимизации: учебное пособие / Новосибирск: Изд-во НГТУ, 2016. – 67 с.

[10] Горчаков А.Ю. ИСПОЛЬЗОВАНИЕ OPENMP ДЛЯ РЕАЛИЗАЦИИ МНОГОПОТОЧНОГО МЕТОДА НЕРАВНОМЕРНЫХ ПОКРЫТИЙ, Перспективные информационные технологии, Самара 14-16 апреля 2018г., издательство Самарского научного центра РАН, 2018, С.613-617

[11] Горчаков А. Ю., Посыпкин М. А. Сравнение вариантов многопоточной реализации метода ветвей и границ для многоядерных систем //Современные информационные технологии и ИТ-образование. – 2018. – Т. 14. – №. 1.

[12] Горчаков А.Ю., Посыпкин М.А., Ямченко Ю.В. экспериментальное исследование трех вариантов реализации метода неравномерных покрытий для многоядерных систем с общей памятью //International Journal of Open Information Technologies. – 2018. – Т. 6. – №. 11 – С. 34-41.

[13] Posypkin M., Usov A. Implementation and verification of global optimization benchmark problems //Open Engineering. – 2017. – Т. 7. – №. 1. – С. 470-478.

[14] Global optimization test functions [Электронный ресурс]: сайт. – https://github.com/alusov/mathexplib (дата обращения: 19.03.2018).

[15] Федеральный исследовательский центр Информатика и управление РАН [Электронный ресурс]: сайт. – Москва: ФИЦ ИУ РАН. – URL: http://hhpcc.frccsc.ru (дата обращения: 12.09.2018)

K-frontal method of nonuniform coverings

A.Y. Gorchakov

***Abstract*— This paper proposes a parallel implementation of the method of non-uniform coverings, designed for computing systems with shared memory. The method of non-uniform coverings is one of the most well-known deterministic methods for solving global optimization problems, based on the scheme of branches and boundaries. Due to the high computational complexity of the method and the wide availability of high-performance multi-core systems with shared memory, the development of parallel implementations of this method is of particular relevance. There are various approaches to parallelizing the method of non-uniform coverings. First, there are several standards for creating multi-threaded applications such as OpenMP, MPI, C ++ 17 functionality. Secondly, there are different approaches to the organization of storage and access to lists of subtasks and synchronization between threads. Thirdly, the branching procedures and evaluation estimates differ. Earlier, a comparison was made of the indicated approaches to the parallelization of the method of non-uniform coverings. One of the most effective methods is the frontal method of non-uniform coverings, it uses a branching strategy wide, OpenMP technology is used for parallelization, and the data storage structure consists of two arrays of pools / subtasks. This approach has the following advantages - ease of implementation, sufficient speed and stability. As disadvantages of the method, you can specify the high requirements for RAM. To eliminate this drawback, this article proposes a parallel implementation of the K-frontal method of non-uniform coverings. The method was tested using a library of test functions on a hybrid high-performance computing cluster of FRC CS RAS.**

***Keywords*— parallel algorithms, method of non-uniform coverings, OpenMP.**

REFERENCES

[1] Yevtushenko Y. G. Chislennyy metod poiska global'nogo ekstremuma funktsiy (perebor na neravnomernoy setke) //Zhurnal vychislitel'noy matematiki i matematicheskoy fiziki, 1971, vol. 6. – pp.1390-1403.

[2] Hansen E., Walster G. W. Global optimization using interval analysis: revised and expanded. – CRC Press, 2003.

[3] Hickernell F. J., Yuan Y. A simple multistart algorithm for global optimization. – 1997.

[4] Martí R., Moreno-Vega J. M., Duarte A. Advanced multi-start methods //Handbook of metaheuristics. – Springer, Boston, MA, 2010. – C. 265-281.

[5] Martí R. et al. Multi-start methods //Handbook of Heuristics. – 2016. – C. 1-21.

[6] Pagès G. The Quasi-Monte Carlo Method //Numerical Probability. – Springer, Cham, 2018. – C. 95-132.

[7] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598), 671-680 (1983).

[8] Lopatin A. S. Metod otzhiga //Stokhasticheskaya optimizatsiya v informatike. – 2005. – T. 1. – №. 1. – S. 133.

[9] P.V. Matrenin, M.G. Grif, V.G. Sekayev, Metody stokhasticheskoy optimizatsii: uchebnoye posobiye / Novosibirsk: Izd-vo NGTU, 2016. – 67 s.

[10] Gorchakov A.Y. ISPOL'ZOVANIYe OPENMP DLYA REALIZATSII MNOGOPOTOCHNOGO METODA NERAVNOMERNYKH POKRYTIY, Perspektivnyye informatsionnyye tekhnologii, Samara 14-16 aprelya 2018g., izdatel'stvo Samarskogo nauchnogo tsentra RAN, 2018, S.613-617

[11] Gorchakov A. YU., Posypkin M. A. Sravneniye variantov mnogopotochnoy realizatsii metoda vetvey i granits dlya mnogoyadernykh sistem //Sovremennyye informatsionnyye tekhnologii i IT-obrazovaniye. – 2018. – T. 14. – №. 1.

[12] Gorchakov A.Y., Posypkin M.A., Yamchenko Y.V. eksperimental'noye issledovaniye trekh variantov realizatsii metoda neravnomernykh pokrytiy dlya mnogoyadernykh sistem s obshchey pamyat'yu //International Journal of Open Information Technologies. – 2018. – T. 6. – №. 11 – S. 34-41.

[13] Posypkin M., Usov A. Implementation and verification of global optimization benchmark problems //Open Engineering. – 2017. – T. 7. – №. 1. – C. 470-478.

[14] Global optimization test functions [Electronic resource]: site. – https://github.com/alusov/mathexplib (дата обращения: 19.03.2018).

[15] Federal Research Center Computer Science and Control of Russian Academy of Sciences [Electronic resource]: site. - Moscow: FRC CS RAS.- URL: http://hhpcc.frccsc.ru (application date: 09/12/2018)"